CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-675

Solving the Maximum Clique Problem Using a Tabu Search Approach

by
Michel Gendreau,
Patrick Soriano
and
Louis Salvail

The authors are grateful to the Quebec Government (Fonds F.C.A.R.) and to the Canadian Natural Sciences and Engineering Research Council (grant 0GP0038816) for their financial support.

Centre de recherche sur les transports — Publication #675

December 1989
Revised June 1990, August 1991, February 1992

92-04607/

Dépôt légal – 2e trimestre 1992

ABSTRACT

We describe two variants of a Tabu Search heuristic, a deterministic one and a probabilistic one, for the maximum clique problem. This heuristic may be viewed as a natural alternate implementation of Tabu Search for this problem when compared to existing ones.

We also present a new random graph generator, the $\hat{p}$-generator, which produces graphs with larger clique sizes than comparable ones obtained by classical random graph generating techniques.

Computational results on a large set of test problems randomly generated with this new generator are reported and compared with those of other approximate methods.

Keywords: Maximum Clique, Tabu Search, Probabilistic Tabu, Random Graph Generator, Approximate Methods.

RÉSUMÉ

Nous décrivons deux variantes, l'une déterministe et l'autre probabiliste, d'une heuristique de recherche avec tabous pour le problème de clique maximale. En regard des autres méthodes de recherche avec tabous pour ce problème, notre approche représente une implantation alternative naturelle.

Nous présentons aussi un nouveau générateur de graphes aléatoires, le générateur $\hat{p}$, qui produit des graphes avec des cliques de plus grande taille que les graphes comparables obtenus par les méthodes de génération de graphes aléatoires classiques.

Nous donnons des résultats numériques pour un ensemble important de problèmes aléatoires et nous les comparons à ceux d'autres méthodes de résolution approchée.

Mots-clés: clique maximale, recherche avec tabous, tabou probabiliste, génération de graphes aléatoires, méthodes de résolution approchée.

1. Introduction

Given a simple undirected graph $G = (V, E)$, a **complete subgraph** is one whose vertices are all pairwise adjacent. A **clique** is a maximal (inclusionwise) complete subgraph and a **maximum clique** is a clique whose cardinality is maximum.

The problem of finding a maximum clique in G is equivalent to the problem of finding a **maximum vertex packing** (independent or stable set of vertices) or a **minimum vertex cover** (set of vertices covering – meeting – all edges) in the complement of G (the graph $\overline{G} = (V, \overline{E})$ where $\overline{E}$ is the complement relative to $V \times V$ of E).

These three problems are important since they occur in a variety of applications: information retrieval systems [12], signal transmission analysis [6, 43], classification theory [11], sociological structures [28, 42], economy [1, 45], timetabling and many others. In timetabling for instance, the size of the maximum clique in the conflict graph of a set of activities is a lower bound on the number of periods required to schedule these activities, while the size of a maximum vertex packing is an upper bound on the number of activities that can be scheduled simultaneously.

With regards to computational complexity, the maximum clique problem (MCP) and its equivalents have been shown to be NP-hard many years ago (see Garey and Johnson [16]). However, there are a number of special classes of graphs for which polynomial algorithms exist (see [2, 26] for a discussion of those). As for arbitrary graphs, a number of non-polynomial algorithms mostly of the branch-and-bound or

enumerative type, have been proposed [2, 3, 4, 5, 9, 10, 13, 18, 21, 31, 33, 34, 38, 39, 40, 41, 44], as well as a few heuristic or approximate methods [17, 30].

In the last ten years or so, Glover [22, 23, 24] and independently Hansen and Jaumard [27] have developed a general metaheuristic for tackling difficult combinatorial optimization problems. This method which is known as **Tabu Search** (TS), or Steepest Ascent/Mildest Descent in Hansen and Jaumard's terminology, combines a local search procedure with a number of clever anti-cycling rules which prevent the search from getting "trapped" in local optima. It has been successfully applied to the Traveling Salesman Problem [23], graph coloring [29], neural networks [46] and flow shop sequencing [47] among others.

Recently, Friden, Hertz and de Werra [14, 15] have developed a TS heuristic for finding stable sets in large graphs (STABULUS) and a TS based exact algorithm for determining the maximum independent set in a graph (TABARIS). Both of these methods have displayed impressive computational results.

In this paper, we describe two variants (a deterministic one and a probabilistic one) of a TS heuristic for the MCP, which may be viewed as a natural alternate implementation of TS for this problem when compared to STABULUS. We also present a new random graph generator, the **$\hat{p}$-generator** which produces test problems with larger clique sizes than comparable problems obtained by classical random graph generation techniques.

The paper is organized as follows. The basic principles of TS are briefly recalled

in section 2. In section 3, we describe in detail our implementation of deterministic TS for the MCP. The probabilistic variant of this algorithm is presented in section 4. In Section 5, we discuss the main differences between our approach and STABULUS and we indicate how we modified STABULUS to transform it into a heuristic for maximum cliques. In section 6, computational results on a large set of randomly generated test problems are reported and compared with those of other approximate methods (simulated annealing-type methods and STABULUS). The $\hat{p}$-generator is also described in that section. Section 7 concludes the paper.

2. Basic principles of tabu search

In this section, we only briefly sketch the basic elements of the Tabu Search metaheuristic. A full description of the method can be found in the fundamental papers of Glover [22, 23, 24] or in the tutorial article by de Werra and Hertz [41]. It should be noted that our presentation closely parallels those of Friden, Hertz and de Werra in [14] and [15] and that mostly the same notation is used.

Given a function f to be maximized over some set X (normally finite), TS starts from some initial feasible point in X and proceeds iteratively from one point in X to another until some termination criterion is met. At that time, one hopes that the optimal solution, or at least a very good approximate solution, has been found.

To each solution s in X, one associates a **set of neighbours** $N(s) \subset X$ which can be readily obtained from s. The basic iterative step then consists in randomly generating a sample $N^*(s)$ of neighbours of s (note that in many applications, it

may be legitimate to take $N^*(s) = N(s)$) and then moving to the best solution s^* in $N^*(s)$ (i.e. $s^* \in \arg \max_{s' \in N^*(s)} \{f(s')\}$).

It should be stressed that since there is no guaranteed strict improvement in the value of the objective function from one iteration to the next, such a procedure may very well cycle. To prevent this from happening, in TS one or several **tabu lists** are introduced. These lists are used to record historical information on the path in X followed by the search procedure during the previous iterations. This information is then used to exclude moves which would tend to make the search process go back to a previously visited solution. Classical examples of tabu lists are: a list of the k last solutions examined (where k is fixed or variable) or one of the k last modifications (changes which occur when we move from a solution s to the next one s^*) that were made (in this case we forbid the reverse modification from occurring). Such types of lists are usually referred to as the **short-term memory** of the search process. It is also possible to incorporate in the method other types of lists which record information on the **intermediate** and **long-term** history of the search. These lists are used respectively to **intensify** the search in an area of the solution space or to **diversify** the search to previously unexplored areas (see [22, 23, 24] for further details on these concepts).

When one uses a tabu list of modifications, rather than previous solutions, it may occur that some "good" solutions may not be considered at some iteration (because the move from the current solution to these is tabu). It may thus be desirable in

certain conditions to cancel the tabu status of a move. One way of accomplishing this is via an **aspiration level** mechanism. This concept is extensively discussed in de Werra and Hertz [46], but for our purposes an illustrative example will suffice. Consider a problem in which the objective function f (to be maximized) may only take integer values. To each possible value z of f, we assign at the start the aspiration level $A(z) = z + 1$. In the course of the search, whenever we consider going from a solution s to a solution s' by a tabu move, we cancel the tabu status of the move if $f(s') \geq A(f(s))$. The aspiration level of $f(s)$ is updated to take the value $f(s') + 1$ after any move from s to s' such that $f(s') \geq A(f(s))$. It is easily seen that even though we allow for the cancellation of the tabu status of some moves, this will not by itself induce cycling since the move from s to s' can only be executed once under tabu status.

We mentioned earlier that the search procedure will keep on going until some termination criterion is met. In most applications, this criterion is simply that the search will stop when a specified number of iterations have been performed without finding a better solution than the best solution currently at hand. When the optimal value of the objective function is known beforehand or has been determined during the search process (in some cases, this may be accomplished through some bounding procedures), the search may obviously be discontinued as soon as a solution with that objective value is found.

3. Deterministic tabu search for the maximum clique selection

In this section, we present a variant of our TS heuristic for the MCP in which no probabilistic elements come into play. In particular, we do not use sampling in the neighbourhoods of solution points.

Basic framework

In many combinatorial optimization problems, the first critical decision which must be addressed when applying TS, is the choice of the solution set to be explored by the search. In the context of the MCP, one should first notice that exploring the set of cliques would be totally impractical, since cliques by themselves are difficult to identify. On the other hand, it is easily seen from basic definitions that a maximum clique is simply a complete subgraph of maximum cardinality. Furthermore, a complete subgraph can be completely described by its vertex-set. If one denotes by X the set of vertex-sets of complete subgraphs in G, then finding the maximum clique in G amounts to solving the optimization problem $\max_{S \in X} f(S) = |S|$.

The set X also lends itself to a nice neighbourhood structure. To better describe it, we first introduce some necessary notation. For $v \in V$, let $B(v)$ be the set of vertices that are adjacent to v in G, i.e. $B(v) = \{v' \in V | (v, v') \in E\}$, and for $S \in X$ denote by $C(S)$ the set of vertices that are adjacent to all vertices in S, i.e. $C(S) = \bigcap_{v \in S} B(v)$. For $S \in X$, also define $N^-(S) = \{S' \in X | S' = S \backslash \{v\}, v \in S\}$ and $N^+(S) = \{S' \in X | S' = S \cup \{v\}, v \in C(S)\}$. $N^-(S)$ is then the set of all complete subgraphs which may be obtained from S by deleting a single vertex

in it, and $N^+(S)$ is the set of complete subgraphs which may be obtained by adding to S a single vertex which is adjacent to all elements of S. By setting $N(S) = N^-(S) \cup N^+(S)$, we thus end up with a consistent neighbourhood structure on X. Also note that $S' \in N^-(S) <=> S \in N^+(S')$, which implies that this neighbourhood structure is symmetric.

It is interesting to remark that this neighbourhood structure has implicitly been exploited by many branch-and-bound algorithms for the MCP (see for instance [18, 40]). We have also used it successfully in a parallel project [20] dealing with the application to the MCP of the simulated annealing method introduced by Kirkpatrick, Gelatt and Vecchi [32].

Another nice feature of this neighbourghhood structure is that, when dealing with augmenting moves, i.e. moves from a solution S to a neighbour S' in $N^+(S)$, the neighbourhood to be considered next is easily determined due to the fact that $C(S') = C(S) \cap B(v')$, where $\{v'\} = S' \backslash S$. When dealing with decreasing moves, i.e. to an S' in $N^-(S)$, however, one must recompute $C(S')$ from scratch.

At this point, we should carefully analyze the basic step in the iterative procedure. Given a current solution S, one notices that $f(S') = |S| + 1$, $\forall S' \in N^+(S)$, and $f(S') = |S| - 1$, $\forall S' \in N^-(S)$. Two implications are to be drawn from this: (1) we should always move to an S' in $N^+(S)$ whenever possible; (2) the objective function gives us no information regarding the specific S' in $N^+(S)$ (or in $N^-(S)$ when no move in $N^+(S)$ is possible) which should be selected. One could arbitrarily select

any S', but such an approach cannot be expected to be successful since it amounts to letting the search wander somewhat aimlessly. A better way of dealing with this problem is to add a secondary criterion to break ties. In our implementation, we have used a **greedy selection rule**: we move to the neighbour for which $|C(S')|$ is maximal. Such a move is the one which has the greatest potential for producing a large clique, since $|S'| + |C(S')|$ is an upper bound on the size of any clique containing S'. When it is applied to $N^+(S)$, this rule is equivalent to the one proposed by Johnson [28] in his greedy heuristic for the MCP, since it amounts to choosing the vertex v' of largest degree in the restricted subgraph $G(S) = (C(S), E(S))$, where $E(S) = \{(v, w) \in E | v \in C(S), w \in C(S)\}$ (see also [16, 17] for further details).

A direct consequence of this choice of tie-breaking rule is that, since we do not use sampling, if we start the search procedure with $S = \emptyset$ (which we do), it will proceed directly to the clique that would be obtained by the application of Johnson's [30] greedy heuristic. This will provide us early with a fairly good solution and sometimes with an optimal one (see [19, 40] for extensive discussions on the quality of the solutions produced by this heuristic).

Tabu lists

In our implementation, we have experimented with two tabu lists. The first one T_1 is simply a list of the last $|T_1|$ solutions visited, while the second T_2 is a list of the last $|T_2|$ vertices that were **deleted.** $|T_1|$ and $|T_2|$ are fixed parameters that are specified as inputs to the procedure.

The list T_1 is used at all times (i.e. whether S' is in $N^+(S)$ or $N^-(S)$), while T_2 is used only when making augmenting moves. The reasons for this unusual scheme are two-fold: (1) it was possible using clever programming tricks to maintain a list of previously visited solutions at reasonable cost; since this allowed for interesting experimentation with the procedure, it was judged to be a valid addition; (2) the set of complete subgraphs is a fairly constrained search space in which the addition of a vertex to the current solution will in many cases orient the search in a drastic way; conversely, removal of a vertex will often "open up" the search; for these reasons, it was decided not to restrict deletions, but only additions. The relative merits of each of these lists will be discussed and analyzed in section 6.

Aspiration level mechanism

With regards to our two tabu list scheme, it should first be noted that no aspiration level mechanism can (and for that matter should) allow for the cancellation of moves which are tabu in T_1, since it is a list of previously visited solutions.

As for moves which are tabu relative to T_2, we may wish to cancel their tabu status using the simple aspiration level function described in the previous section. For our specific application, this mechanism can be effectively implemented in an implicit form by adding a test which will allow us to move to a solution S' if $f(S') > z^*$ where z^* is the size of the largest clique found so far.

To see why this test is the only one required to implement this mechanism, consider the following facts:

- since at the start of the procedure one has $z^* = 0$, and since the objective function at most increases by 1 in any iteration, if the current value of z^* is k, this implies that z^* must have taken all the integer values in the range $[0, k]$ in previous iterations; one must therefore have visited solutions for which an improvement was found for every value of f between 0 and $(k-1)$, which implies that $A(z) = z + 2,\ z = 0, 1, ..., k-1$;
- in an augmenting move, one always has $f(S') = f(S) + 1$, which implies that $f(S') < A(f(S))$, unless $f(S) = k = z^*$, i.e. unless $f(S') > z^*$.

Re-directing search

As stated earlier, given any S' in X, $|S'| + |C(S')|$ is an upper bound on the size of a clique containing S'. If during the search process, we end up at a solution S such that $|S'| + |C(S')| \leq z^*, \forall S' \in N^+(S) \backslash T_1$, this means that there is no hope of finding a better clique through any non-tabu augmenting move; it is therefore advantageous to immediately re-direct the search to some other portion of the solution space. If there exists non-tabu solutions in $N^-(S)$, we choose the best among these according to our tie-breaking rule, otherwise we arbitrarily choose any element in $N^-(S)$ to continue the search from.

Termination criterion

We terminate the procedure when *MaxIter* iterations without improvement in z^* occur. *MaxIter* is an input parameter to the procedure.

4. The probabilistic variant

We have also experimented with a probabilistic variant of the previous procedure. It differs from its deterministic counterpart by the following features:

(1) If a current solution S is a local maximum (i.e. if $N^+(S) = \emptyset$), the procedure randomly removes from S $\max\{k, |S|\}$ vertices where k is an input parameter; the search then proceeds from the new solution thus obtained.

(2) If $N^+(S) \neq \emptyset$, a random sample $N^*(S)$ of size $\max\{L, |N^+(S)|\}$, where L is also an input parameter, is generated; if $N^*(S)$ contains at least one element S' such that moving to S' is non-tabu or S' allows for improvement of z^*, the search proceeds to the element of $N^*(S)$ satisfying these conditions which is best according to the tie-breaking rule. Otherwise, a sequence of deleting moves is performed as in (1) above.

The second feature of this variant is simply the introduction of sampling for augmenting moves as described in section 2 above. The other feature, which more closely resembles a "random shake-up" of the solution than anything else, has been motivated by our specific application: as mentioned earlier, computing $C(S')$ after a deletion is much more expensive since it must be done from scratch; it is therefore advantageous to perform multiple deletions at the same time to cut down on these computations. We must mention however that preliminary computational results have indicated that finding the correct value of L for a given problem size is a tricky affair.

In our implementation of the probabilistic variant, the search is started from a

solution provided by the greedy heuristic, instead of starting from an empty set. This procedure is equivalent to introducing sampling only after the first local maximum has been found.

5. Transforming STABULUS to find maximum cliques

As mentioned in the introduction, STABULUS [14] is a TS heuristic for finding stable (independent) sets in large graphs. It thus solves a problem which is equivalent to that of finding cliques. However, it should be noted right away that, contrary to our method, STABULUS is not strictly speaking an optimizing procedure: STABULUS will search a graph for a stable set of size k, where k is a **specified input parameter**, instead of trying to determine a maximum independent set. This difference in objective has important consequences which we shall now examine.

Knowing beforehand the cardinality of the stable set being searched for suggests the exploration of a different solution space: the set of subsets of k vertices. In fact, STABULUS moves from a subset which is "almost" an independent set to another, until a stable set (the equivalent of a complete subgraph in clique terminology) is found, at which time the procedure can be immediately terminated since the desired solution has been found. One may thus interpret STABULUS as a "dual-type" procedure, while our heuristics which move in the space of feasible solutions (complete subgraphs) correspond to a "primal" approach.

If one wants to use STABULUS to find a maximum independent set, and therefore solve a problem equivalent to the one we are tackling, the size of such a set must

first be "guessed" correctly. It is well-known that, for certain types of graphs, the cardinality of the maximum independent set can be readily estimated from theoretical analysis, but this is obviously not the case for arbitrary graphs, which implies that this approach is not viable when dealing with "unknown" graphs. To circumvent this difficulty, we have developed an optimizing variant of STABULUS which we call "Iterated-Stabulus". This procedure can be summarized as follows. The greedy heuristic is first applied, yielding an independent set of size $\tilde{k}$. As was discussed in the previous sections, $\tilde{k}$ will in general be fairly close to the desired solution. We then apply STABULUS iteratively with $k = \tilde{k}+1, \tilde{k}+2, ...$ The procedure terminates when STABULUS fails to find a stable set of a given size k within a certain number of iterations or when a CPU time limit on the overall procedure is reached.

As a final remark, it should be repeated that "Iterated-Stabulus" can be used to determine the maximum clique of a graph G by simply considering the complement graph $\overline{G}$ of G in the STABULUS steps of the procedure.

6. Computational results

To test our procedure and to experiment with various settings for the input parameters, we have developed three programs which correspond respectively to the deterministic version with a single tabu list (list T_1), the same with two tabu lists, and the probabilistic version.

To allow for extensive testing of the programs, we decided to use randomly generated problems. This has been a widely used practice in the past. In fact, for a

number of years, most authors have tested their algorithms with problems generated by the **uniform random graph scheme**. This method works with two input parameters: a vertex-set size n and a density p. It consists in examining each of the possible $\binom{n}{2}$ edges in the graph in sequence and adding them to the graph with probability p. Unfortunately, such graphs have been studied by researchers in random graph theory and it is now well-known that they possess very strong properties with regards to many attributes, one of them being the size of the maximum clique which most often falls in a very narrow range for a given choice of n and p (see [7, 8, 35, 36, 37] for detailed results on this subject). To generate a more diverse sample of graphs, we thus decided to develop a new random graph generator, the $\hat{p}$-generator, which works with three rather than two inputs: n which is the vertex-set size as before, and a and b, two real numbers which satisfy $0 \leq a \leq b \leq 1$.

***Procedure* $\hat{p}$-*generator* (n, a, b)**

begin

for $i := 1$ to n do $\hat{p}[i] :=$ uniform (a, b);

for $i := 1$ to $(n-1)$ do

for $j := (i+1)$ to n do

generate edge (i, j) with probability $\frac{\hat{p}[i]+\hat{p}[j]}{2}$

end.

Note that **when $a = b$**, one has $\hat{p}[i] = a$ for all vertices and **the $\hat{p}$-generator is**

then equivalent to the classical uniform random graph generator. The $\hat{p}$-generator can thus be seen as a generalization of the uniform generator.

The following properties apply in general to $\hat{p}$-generated graph:

(1) their expected density $\overline{p}$ is equal to $(a+b)/2$;

(2) for any vertex v_i, the conditional expected degree of v_i given $\hat{p}[i]$ is $d(v_i|\hat{p}[i]) = \left(\frac{\hat{p}[i]}{2} + \frac{a+b}{4}\right)(n-1)$, which means that degree spread will increase rapidly with the difference $(b-a)$;

(3) because of the larger degree spread, one would expect, for any fixed $\overline{p}$, the size of the maximum clique to increase with the difference $(b-a)$; this intuitive assertion is confirmed by our empirical results.

A total of 180 test problems generated with the $\hat{p}$-generator were solved. For each of the three problem sizes, n=100, 300, 500 vertices, 6 series of 10 problems corresponding to different settings of the a and b parameters were considered. This provided us with 6 families of random graphs: 3 of the classical type (when $a = b$) with expected densities of p = 0.25, 0.5, and 0.75, respectively, and 3 others of the new type (with $a < b$) with the same expected densities but presenting significant expected vertex degree spreads.

In preliminary testing of the two tabu methods, we experimented with various settings of the input parameters. These ranged for the size of the T_1 list from 0 to 150 and for that of T_2 from 0 to 40. In the deterministic tabu, setting $|T_2| = 0$ (i.e single-list tabu with T_1 as the tabu list) provided very satisfactory results qualitywise.

On the other hand, with $|T_1| = 0$ (i.e. single-list tabu with only T_2) the quality of the solutions obtained was clearly inferior to that when $|T_1| > 0$. These observations also hold true for the probabilistic variant. Furthermore, for this second approach several ways of determining the values of parameters k and L were tested but no one method stood out as being superior. The only guidelines that could be inferred were that these parameters should take values big enough so as to promote a "wider" exploration of the solution space but not too much, thus preventing unnecessary "jumps" of the search process.

In those experiments with parameter settings for the two tabu methods, we found out that the size of list T_1 must not be too large and that list T_2 must be fairly small (5 elements), otherwise the quality of the solutions starts to decrease. Interestingly enough, the "ideal" size of the lists seems to be independent of problem size, graph density and vertex degree spread. In some cases, it is possible to slightly improve the solutions by using a larger value of *MaxIter*, but the execution times are then much higher. It should be noted that since the probabilistic tabu performs sampling, it will run much faster on denser graphs ($\overline{p} \geq 0.5$). This makes it possible (and, to a certain extent, necessary) to use a larger value of *MaxIter* than in the deterministic variant.

The "optimal" values for the parameter settings for each of the three methods can be found in Table 1 under the columns ST ("Single-list Tabu"), DT ("Double-list Tabu") and PT ("Probabilistic Tabu"). These values of $|T_1|$ and $|T_2|$ are the ones that were used in the final computational experiments.

The same problems were also solved with a Simulated Annealing type procedure based on the "Rejectionless Method" of Greene and Supowit [25]. This method has been specially customized to deal with the MCP. In a parallel project on the application of Simulated Annealing to the MCP it was found to be the best performing of all tested methods [20].

In order to provide a fair comparison of all methods, the stopping criterion used in the final computational tests was changed from *MaxIter* to *TimeLimit* (i.e. the search is stopped after a fixed amount of CPU time has been expended). The values chosen for *TimeLimit* correspond in general to values of *MaxIter* in the range 200 to 400, as determined by the preliminary testing. During the same time span, the "Rejectionless Method" (RM in the Tables) will almost complete a cooling schedule made up of 29 "temperature" levels; at each level t, the "acceptance probability" is $P_t = (0.9)^t$ and the level length is $\ell_t = 200 \times (1.05)^{t-1}$. A more detailed description of this cooling schedule and the RM program can be found in [20].

All programs were implemented by the same programmer using the same language (PASCAL), compiler (TURBO PASCAL), and basic graph manipulation routines.

The computational results are presented in Tables 2 and 3 respectively for graphs of the classical type ("uniform graphs") and for the new type with large vertex degree spreads ("extreme graphs"). Average solution sizes for each group of 10 problems obtained after spending a specified amount of CPU time (under column "Time") are indicated in the Tables for each of the four tested methods. Note that the solution

sizes are given not only for the maximum CPU time allotted (*TimeLimit*), but also for intermediate times. This allows us to get a better understanding of the aggressiveness of the methods in moving towards the solution. All CPU times are in seconds for an IBM PS/2 Model 70 with a 20 MHz 80386 CPU running under DOS.

In Table 2, the last column ($\hat{\alpha}$) indicates the most probable clique size for a given pair (n, p). For $p = 0.5$, the values have been taken directly from the paper on STABULUS [14], while for $p = 0.25$ and 0.75 the values have been determined from a result of Matula (which can be found in [37]). It should be stressed that $\hat{\alpha}$ does not necessarily coincide with the average optimal clique size for a specific group of test problems, and that for $p = 0.75$, $\hat{\alpha}$ is not as precise an estimator of the maximum clique size as for sparser graphs.

We will now briefly comment on the results of Tables 2 and 3.

First, one can easily see that the average size of solutions for a fixed density $\overline{p}$ is always greater when $a < b$ than when $a = b$, and that the difference between them increases with the spread $(b - a)$. This confirms the intuitive assertion we made at the beginning of this section. Hence, the objective that was initially set forward when the $\hat{p}$-generator was conceived has effectively been attained.

The second comment one can make is that the solution times required by the extreme graphs are much higher than those for the uniform graphs. This can be directly linked to the fact that those graphs display much larger cliques, especially for the large problem sizes.

If one compares the results produced by the various methods after the specified *TimeLimit*, it is difficult to determine a clear winner. RM, ST and DT seem to have a roughly equal performance, while PT has a slight advantage overall. This advantage is more pronounced when one restricts its attention to the larger graphs with high densities. In these cases, it seems that the combination of randomness in the search induced by the sampling and "random shake-up" features of PT with the purposefulness of Tabu Search yields the best of two worlds.

If one looks at the speed at which the methods move towards good solutions, however, the picture is totally different. The three Tabu variants produce large cliques very quickly (in relative terms), but RM is outclassed. This is very important if one values speed over optimality of the solution.

To determine the quality of the solutions produced by our heuristic procedures, 6 of 18 problem series (those for $n = 100, \overline{p} = 0.25, 0.5$ and $n = 300, \overline{p} = 0.25$) were solved exactly using the implicit enumeration algorithm of Gendreau, Picard and Zubieta [18]. 6 other problem series with $n = 50$ and $\overline{p} = 0.25, 0.5, 0.75$ were also solved with this exact method and with PT (the *Timelimit* for these runs was set at 10 seconds). The average clique size and the number of times that the optimal solution was found for each of these series both for PT and for the greedy heuristic are reported in Table 4. These results clearly show that the solutions produced by PT (and, by way of consequence, by our other Tabu Search procedures) are optimal for a large proportion of these problems, 114 out of 120 instances or 95% of the time, and

that when the optimal solution is not found, the best clique found by PT is just one vertex smaller that the optimum. In comparison, the greedy heuristic finds the optimal solution in only 54 problems out of 120, or 45% of the time, and on several occasions the solution it finds is off the optimal by two vertices or more (10 times or 8.3%).

To allow for comparison of our method with STABULUS, we solved the 60 500–vertex problems (the most significant in our mind) with the Iterated-Stabulus procedure described in the previous section, using the same *TimeLimit* termination criterion that had been set for the other heuristics. The length of the tabu lists for the STABULUS steps was determined from the formulas in the original STABULUS article [14]: $|T_1| = 27 + \hat{\alpha}(n - \hat{\alpha})/120$, $|T_2| = |T_3| = 2$, where $\hat{\alpha}$ is the most probable clique size for a uniform graph of a given density. Note that for each density the same value of $|T_1|$ was used for both types of graphs, even though the average clique size is much larger for extreme graphs (experiments with different values of $|T_1|$ yielded smaller average clique sizes for these graphs).

The computational results obtained with the Iterated-Stabulus procedure and with our most efficient heuristic PT are summarized in Table 5. In this Table, the "Greedy" column indicates the average clique sizes found by the greedy heuristic before starting the STABULUS steps (the time required by this step can also be found under the heading "Time").

An illustration of the comparative behaviour of the various methods tested for the 500–vertex graphs with $[a, b] = [0.75, 0.75]$ (uniform graphs) and $[a, b] = [0.5, 1.0]$ (extreme graphs) is given respectively in Figures 1 and 2 which depict average solution

size evolution with respect to elapsed CPU time.

A few comments are now in order.

1) With regards to the implementation of Iterated-Stabulus, it must be noted that, for all graphs, the greedy heuristic provides very quickly an excellent starting point from which to apply the STABULUS steps, which means that only a few values of k have to be considered in these steps. It is thus reasonable to assert that Iterated-Stabulus provides for a fair comparison of the STABULUS approach with ours when dealing with arbitrary graphs.
2) Given enough CPU time, Iterated-Stabulus and PT both succeed to produce large cliques for all types of graphs studied. However, it must be noted that PT is a more aggressive method in the sense that it yields results comparable to those of Iterated-Stabulus in about half the CPU time. This might become critical in situations where the time available is limited, for instance when maximum clique problems are solved repeatedly as subproblems of a larger problem.
3) Before performing the comparison between Iterated-Stabulus and PT, we thought that Iterated-Stabulus would perform relatively better on the uniform random graphs and PT on the extreme random graphs. Our empirical results do not support at all these intuitive beliefs.

As a final remark, it should be stressed that with regards to comparisons with other methods reported in the literature, our results were obtained on a micro-computer and that our CPU times cannot thus be compared directly with execution times on

mainframe computers.

7. Conclusion

We have presented two variants of a Tabu Search approach for the maximum clique problem. With proper parameter settings, these two variants have been shown to be able to produce very good solutions in a reasonable amount of CPU time on a micro-computer. When compared to other approximate methods for the MCP, our Tabu algorithms are competitive with the best of these.

We have also proposed and implemented a new random graph generator which allows users to produce more varied graphs than classical techniques at just a slight increase in complexity and CPU time. This generator can be expected to become widely used for testing MCP algorithms in the coming years.

8. Acknowledgements

We thank D. de Werra and A. Hertz for providing us with a copy of STABULUS.

9. References

[1] G. AVONDO-BODENO, Economic Applications of the Theory of Graphs, Gordon & Breach Science Publishers, New York, 1962.

[2] L. BABEL, Finding maximum cliques in arbitrary and in special graphs, Report TUM-M9008, Mathematiches Institut und Institut für Informatik, Technische Universität München, 1990. To appear in **Computing**.

[3] L. BABEL and G. TINHOFER, A Branch and Bound Algorithm for the Maximum Clique Problem, **ZOR-Methods and Models of Operations Research 34** (1990), 207–217.

[4] E. BALAS and J. Xue, Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs, **SIAM J. Comput.** Vol. 20, No. 2 (1991), pp. 209–221.

[5] E. BALAS and C.S. YU, Finding a maximum clique in an arbitrary graph, **SIAM J. Computing 15** (1986), 1054-1068.

[6] C. BERGE, Théorie des graphes et ses applications, Dunod, Paris, 1962.

[7] B. BOLLOBAS, Random Graphs, Academic Press, London, 1985.

[8] B. BOLLOBAS and P. ERDÖS, Cliques in random graphs, **Math. Proc. Camb. Phil. Soc. 80** (1976), 419-427.

[9] C. BRON and J. KERBOSCH, Finding all cliques of an undirected graph, **Comm. ACM 16** (1973), 575-577.

[10] R. CARRAGHAN and P.M. PARDALOS, An exact Algorithm for the Maximum Clique Problem, **Operations Research Letters 9** (Nov. 1990), pp. 375–382.

[11] V. DEGOT and J.M. HUALDE, De l'utilisation de la notion de clique (sous-graphe complet symétrique) en matière de typologie des populations, **Revue française d'automatique, informatique et recherche opérationnelle: Recherche opérationnelle 9** (1975), 5-18.

[12] N. DEO, Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Englewood Cliffs, 1974.

[13] C. EBENEGGER, P.L. HAMMER, and D. DE WERRA, Pseudo-Boolean functions and stability of graphs, **Annals of Discrete Mathematics 19** (1984), 83-98.

[14] C. FRIDEN, A. HERTZ, and D. DE WERRA, Stabulus: a technique for finding stable sets in large graphs with tabu search, **Computing 42** (1989), 35-44.

[15] C. FRIDEN, A. HERTZ, and D. DE WERRA, Tabaris: An exact algorithm based on Tabu Search for finding a maximum independent set in a graph, **Computers and O.R. 17** (1990), 437–445.

[16] M.R. GAREY and D.S. JOHNSON, Computers and Intractibility: a Guide to the the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[17] M. GENDREAU, A fast greedy algorithm for the maximum clique problem, paper presented at the TIMS/ORSA Meeting, New Orleans, May 1987.

[18] M. GENDREAU, J.-C. PICARD, and L. ZUBIETA, An efficient implicit enumeration algorithm for the maximum clique problem, in Advances in Optimization and Control (H. A. Eiselt and G. Pederzoli, Eds.), Springer-Verlag, Berlin, 1988.

[19] M. GENDREAU, L. SALVAIL, and P. SORIANO, An appraisal of greedy heuristics for the maximum clique problem, Centre de recherche sur les transports, Université de Montréal, forthcoming.

[20] M. GENDREAU, P. SORIANO, and L. SALVAIL, Simulated annealing and cliques, Centre de recherche sur les transports, Université de Montréal, forthcoming; paper presented at the ORSA/TIMS Meeting, New York, October 1989.

[21] L. GERHARDS and W. LINDENBERG, Clique detection for nondirected graphs: two new algorithms, **Computing 21** (1979), 295-549.

[22] F. GLOVER, Future paths for integer programming and links to artificial intelligence, **Computers and O.R. 13** (1986), 533-549.

[23] F. GLOVER, Tabu Search Part I, **ORSA Journal on Computing 1** (1989), 190-206.

[24] F. GLOVER, Tabu Search Part II, **ORSA Journal on Computing 2** (1990), 4-32.

[25] J.W. GREENE and K.J. SUPOWIT, Simulated annealing without rejected moves, **IEEE Trans. Computer Aided Design CAD-5** (1986), 221-228.

[26] M. GRÖTSCHEL, L. LOVASZ, and A. SCHRIJVER, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, Berlin, 1988.

[27] P. HANSEN and B. JAUMARD, Algorithms for the maximum satisfiability problem, RUTCOR Research Report 43-87, Rutgers University, 1987.

[28] F. HARARY, Graph theory as a structural model in the social sciences, in Graph Theory and its Applications (B. Harris, ed.), Academic Press, New York, 1970.

[29] A. HERTZ and D. DE WERRA, Using Tabu Search techniques for graph coloring, **Computing 29** (1987), 345-351.

[30] D.S. JOHNSON, Approximation algorithms for combinatorial problems, **Journal of Computer and Systems Sciences 9** (1974), 256-278.

[31] D.S. JOHNSON, M. YANNAKAKIS, and C.H. PAPADIMITRIOU, On generating all maximal independent sets, **Information Processing Letters 27** (1988), 119-123.

[32] S. KIRKPATRICK, C.D. GELATT Jr., and M.P. VECCHI, Optimization by simulated annealing, **Science 220** (1983), 671-680.

[33] E. LOUKAKIS, A new backtracking algorithm for generating the family of maximal independent sets of a graph, **Computer and Mathematics with Applications 9** (1983), 583-589.

[34] E. LOUKAKIS and C. TSOUROS, Determining the number of internal stability of a graph, **International J. Computer Math 11** (1982), 207-220.

[35] D.W. MATULA, The employee party problem, **Not. A.M.S. 19** (1972), A-382.

[36] D.W. MATULA, The largest clique in a random graph, Technical Report CS7608, Southern Methodist University, 1976.

[37] D.W. MATULA, Expose-and-merge exploration and the chromatic number of a random graph, **Combinatorica 7** (1987), 275-284.

[38] P.M. PARDALOS and N. DESAI, An Algorithm for Finding a Maximum Weighted Independent Set in Arbitrary Graph, **Inter. J. Computer Math. 38** (1991), pp. 163–175.

[39] P.M. PARDALOS and A. PHILLIPS, A Global Optimization Approach for Solving the Maximum Clique Problem, **Intern. J. Computer Math. 33**, (1990), pp. 209–216.

[40] P.M. PARDALOS and G.P. RODGERS, A branch and bound algorithm for the maximum clique problem, to appear in **Mathematical Programming**.

[41] J.M. ROBSON, Algorithms for the Maximum Independent Sets, **J. Of Algorithms 7** (1986), pp. 425–440.

[42] B. ROY, Algèbre moderne et théorie des graphes, Tome 1, Dunod, Paris, 1969.

[43] C.E. SHANNON, The zero-error capacity of a noisy channel, Composium on Information Theory, I.R.E. Transactions 3 (1956).

[44] R.E. TARJAN and A.E. TROJANOWSKI, Finding a maximum independent set, **SIAM J. on Computing 6** (1977), 537-546.

[45] J. TURNER and W.H. KAUTZ, A survey of progress in graph theory in the Soviet Union, SIAM 12 (1970).

[46] D. DE WERRA and A. HERTZ, Tabu Search techniques: A tutorial and application to neural networks, **OR Spektrum 11** (1989), 131-141.

[47] M. WIDMER and A. HERTZ, A new heuristic method for solving the flow shop sequencing problem, **European Journal of Operational Research 41** (1989), 186-193.

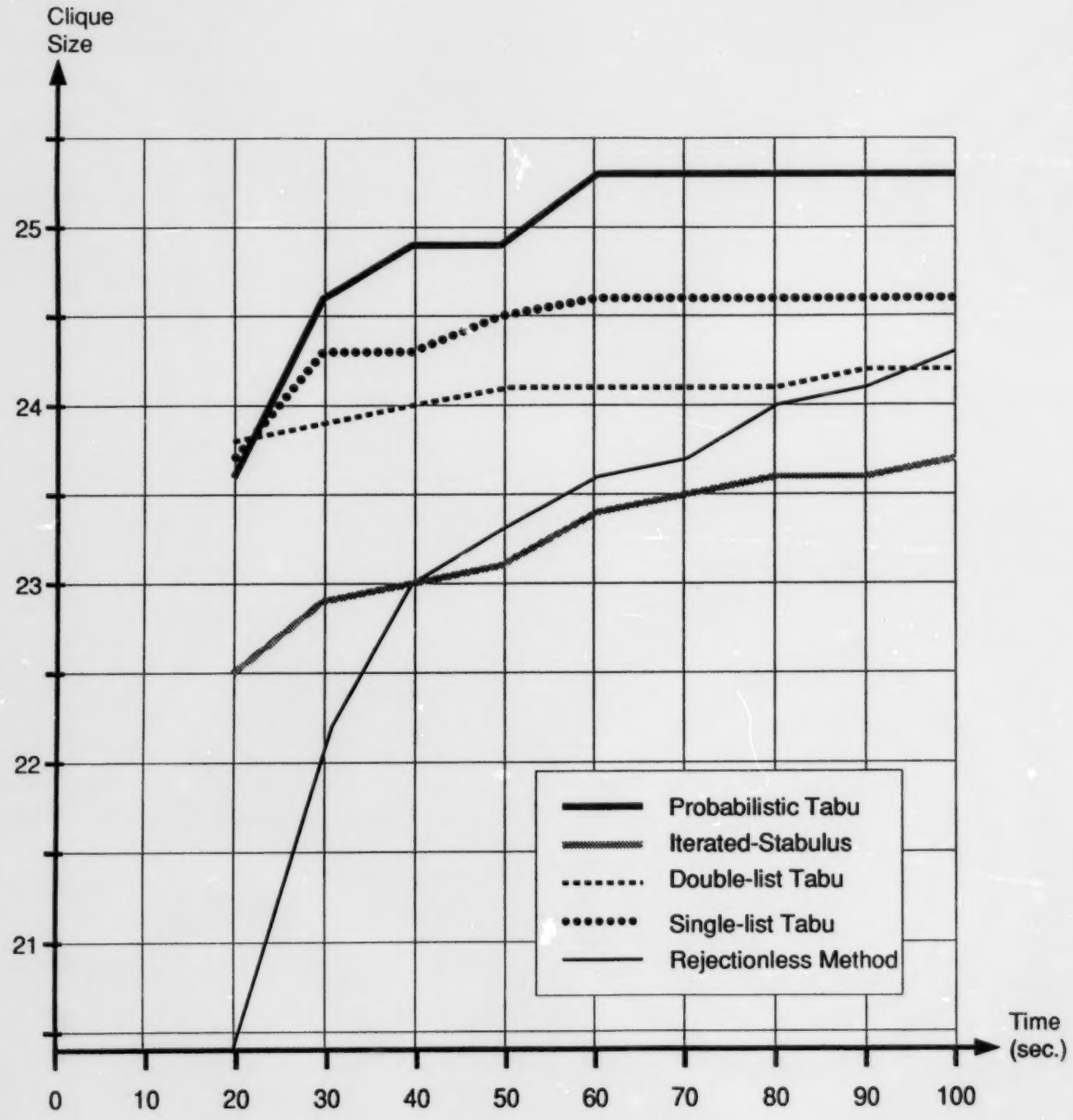

Figure 1 : Average clique size over time for random graphs with n=500 and $\bar{p}$=0.75[0.75,0.75]

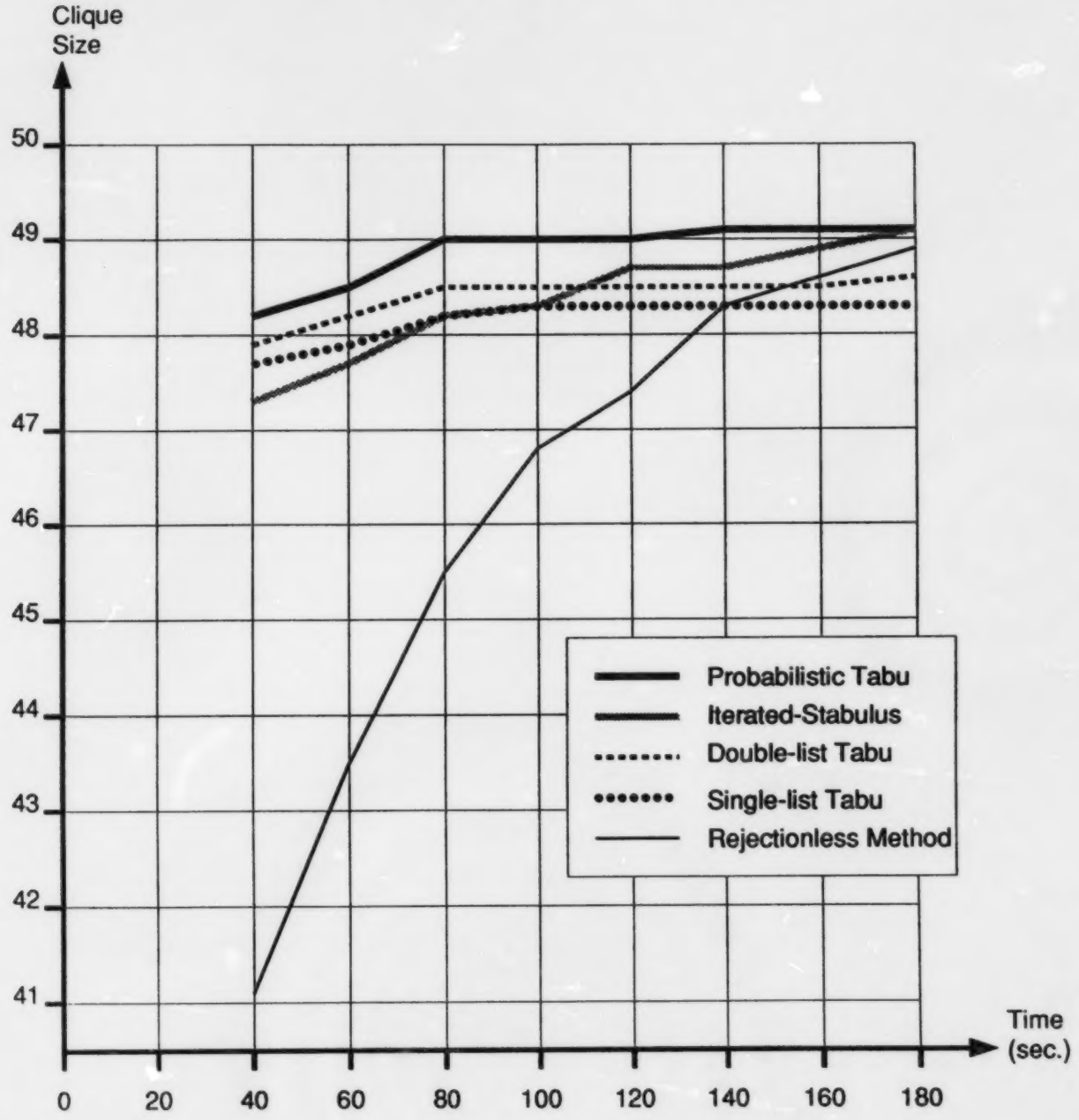

Figure 2 : Average clique size over time for random graphs with n=500 and $\bar{p}$=0.75[0.5,1.0]

Parameter	ST	DT	PT
|T1|	100	100	150
|T2|	---	5	5
MaxIter	250	250	300

Table 1 : Parameter settings

n	p	Time	RM	ST	DT	PT	$\hat{\alpha}$
100	0.25	6	5.6	5.6	5.5	5.6	6
		12	5.7	5.7	5.6	5.7	
	0.50	10	8.8	9.1	9.1	9.0	9
		20	9.2	9.1	9.2	9.1	
	0.75	10	15.3	16.6	16.6	16.6	15
		20	16.7	16.6	16.6	16.7	
300	0.25	9	6.3	6.3	6.6	6.2	7
		15	6.6	6.6	6.7	6.6	
		30	6.9	6.6	6.9	6.7	
	0.50	9	10.0	11.1	11.2	10.9	12
		15	10.7	11.1	11.2	11.1	
		30	11.5	11.3	11.2	11.1	
	0.75	30	19.9	21.4	21.8	21.4	22
		50	21.7	21.8	22.1	21.7	
		100	22.3	21.8	22.3	22.3	
500	0.25	9	6.6	7.1	7.1	6.9	8
		15	7.0	7.1	7.1	7.1	
		30	7.0	7.1	7.2	7.1	
	0.50	15	11.4	12.3	12.2	12.0	13
		25	12.1	12.3	12.2	12.2	
		50	12.2	12.3	12.5	12.7	
	0.75	30	22.2	24.3	23.9	24.6	25
		50	23.3	24.5	24.1	24.9	
		100	24.3	24.6	24.2	25.3	

Table 2 : Computational results for uniform random graphs

n	$\overline{p}$	[a-b]	Time	RM	ST	DT	PT
100	0.25	0.0-0.5	6	6.2	6.4	6.4	6.3
			12	6.4	6.4	6.4	6.4
	0.50	0.0-1.0	10	15.2	15.7	15.6	15.6
			20	15.5	15.7	15.7	15.7
	0.75	0.5-1.0	10	19.8	21.8	21.8	21.8
			20	21.4	21.8	21.8	21.8
300	0.25	0.0-0.5	9	7.8	8.0	8.1	8.1
			15	7.9	8.0	8.2	8.1
			30	8.1	8.0	8.2	8.1
	0.50	0.0-1.0	15	22.8	25.4	25.7	25.3
			25	24.4	25.6	25.8	25.6
			50	25.6	25.6	25.9	25.9
	0.75	0.5-1.0	30	34.2	37.6	37.8	37.7
			50	36.2	37.7	37.8	37.9
			100	38.1	37.9	38.0	38.2
500	0.25	0.0-0.5	9	8.1	8.9	8.9	8.6
			15	8.5	9.0	9.0	8.8
			30	9.0	9.0	9.0	8.9
	0.50	0.0-1.0	30	30.0	33.8	33.8	33.9
			50	32.8	34.0	33.9	34.1
			100	34.0	34.0	33.9	34.2
	0.75	0.5-1.0	60	43.5	47.9	48.2	48.5
			120	47.4	48.3	48.5	49.0
			180	48.9	48.3	48.6	49.1

Table 3 : Computational results for extreme random graphs

n	$\bar{p}$[a,b]	Greedy		Probabilistic Tabu		Exact	$\hat{\alpha}$
		Average	# Opt	Average	# Opt		
50	0.25[0.25,0.25]	4.4	6/10	4.8	10/10	4.8	5
	0.25[0.00,0.50]	4.5	4/10	5.1	10/10	5.1	---
	0.50[0.50,0.50]	7.4	6/10	7.8	10/10	7.8	8
	0.50[0.00,1.00]	9.7	8/10	9.9	10/10	9.9	---
	0.75[0.75,0.75]	12.4	4/10	13.2	10/10	13.2	12
	0.75[0.50,1.00]	15.2	8/10	15.4	10/10	15.4	---
100	0.25[0.25,0.25]	5.0	4/10	5.7	10/10	5.7	6
	0.25[0.00,0.50]	5.6	4/10	6.3	9/10	6.4	---
	0.50[0.50,0.50]	8.3	4/10	9.1	9/10	9.2	9
	0.50[0.00,1.00]	14.9	3/10	15.7	10/10	15.7	---
300	0.25[0.25,0.25]	5.9	0/10	6.7	7/10	7.0	7
	0.25[0.00,0.50]	7.3	3/10	8.1	9/10	8.2	---

Table 4 : Comparison of solutions produced with the Greedy heuristic, Probabilistic Tabu and an exact algorithm.

$\overline{p}$ [a-b]	Time	PT	Iterated-Stabulus	
			Greedy	Stabulus
0.25 [0.25-0.25]	0.8		6.3	
	9	6.9		6.6
	15	7.1		6.6
	30	7.1		6.9
0.50 [0.50-0.50]	1.3		11.3	
	15	12.0		11.6
	25	12.2		11.8
	50	12.7		12.0
0.75 [0.75-0.75]	2.7		22.5	
	30	24.6		22.9
	50	24.9		23.1
	100	25.3		23.7
0.25 [0.00-0.50]	0.8		7.9	
	9	8.6		8.2
	15	8.8		8.5
	30	8.9		8.7
0.50 [0.00-1.00]	3.3		32.6	
	30	33.9		33.4
	50	34.1		33.9
	100	34.2		34.3
0.75 [0.50-1.00]	6.5		46.5	
	60	48.5		47.7
	120	49.0		48.7
	180	49.1		49.1

Table 5 : Comparison of Probabilistic Tabu (PT) and Iterated-Stabulus for graphs of 500 vertices